Deep Pollination Challenge

Team Ecologists

Mohammad Lansari, Thibaut Soulard, Ihsan Ullah, Xavier Bou, Vaibhav Arora, Eric Santiago

Github 1 : https://github.com/Ecologists/deep_pollination
Codalab 1 : https://competitions.codalab.org/competitions/28635
Github 2 : https://github.com/Ecologists/deep_pollination_raw
Codalab 2 : https://competitions.codalab.org/competitions/28996
Video : https://youtu.be/T8ALa9phYGY

I. Background

Insects, specially pollinating insects have a great role in the biodiversity in the nutrient cycle and the functioning of ecosystems [1]. One of the the main threats to our ecosystem is the continuous decline in the population of various pollinating insects. About 75% of the plant species require insect pollinators and of which, bees are the primary pollinators. Other major pollinators are pollen wasps, ants, butterflies, moths and flies. Pollination is a ecosystem process that is essential to support the production of a wide range of crops and to sustain some essential food-chains in the ecosystem. This process is increasingly under threat, as a consequence of the loss of habitats and populations of these pollinators by increasing use of pesticides and insecticides. Directly impacted are the pollinator-dependent industries and agriculture as a whole [2].

Monitoring the population of pollinating insects has traditionally been done manually. Machine learning and in particular, deep learning techniques offer new possibilities to do this task with more speed, accuracy and efficiently. Therefore, image recognition techniques to spot pollen carrying insects is of particular interest. A first step to establish such a classification task is to actually recognize the type of insect in the image.

Therefore in this challenge, we address the supervised learning problem on image of insects. The nature of this problem is very challenging attributed to high varying texture, image background and possible bias in data due to non-uniform collection sources. Moreover, small scaled objects have been proved to have lower accuracy in general (see Figure 1).

The results for the existing state of the art for insect classification discussed in [4] show use of traditional feature extraction with machine learning classifiers as shown in Figure 2 for small image data gathered in lab settings. To the best of our knowledge, the big deep learning attempt to tackle the problem was from the challenge organized on the **RAMP** platform [14] to classify images of insects from

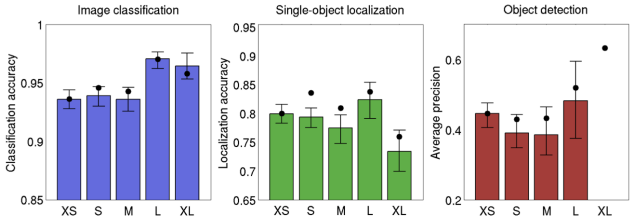


Figure 1. Classification accuracy in regards to natural object size (from [3])

the SPIPOLL crowdsourcing project of the Paris Museum of Natural History, with the scientific goal of quantitatively studying pollinating insects in France. With 70K training images and 403 classes with the bottom 100 classes having less than ten images per class, about 90% accuracy of top class prediction was achieved.

	[62]	[47]	[28]	[52]	[25]
Least Square	84.2	83.3			
LDA				64.0	
SVM	88.4		**85.0**	60.0	**92.0**
Decision tree			58.3	36.0	
MLP				58.5	76.0
Bayes		**89.9**	65.9	**65.2**	
Logistic				63.2	
Parzen density	86.3	84.7			
Random forest			83.2		
***k*-NN**	77.4	83.8	71.6		
Nearest Mean	**89.5**	63.6			

Figure 2. Insect classification results for images in lab based settings from [4]

On similar line, but with a much larger dataset of images from generic settings too, this challenge seeks to **explore limits** of insect classification by using deep learning, specifically learn weights using transfer learning methods on insect classification which can further be reused for similar tasks. A novel boundary we want the challengers to explore by the means of this competition is to gain insights into the model and the data likewise by using Explainable AI (XAI).

II. Material and methods

The project consists of two main tasks :

- classify images into 5 classes (multi-class classification) with deep-learning models and transfer learning
- explain on which features the algorithm has made its decision (Explainability)

The dataset used in this project is a combination of two datasets:

- Bee vs wasp dataset from Kaggle (11,415 images). [Figure 3]
- Dataset from Museum National d'histoire Naturelle (210,676 images) [Figure 4]

Table I. Dataset Statistics

Dataset	Bee	Wasp	Insect	Other	Total
Kaggle	3182	4941	2439	853	11,415
Museum	74413	10796	96106	29361	210,676

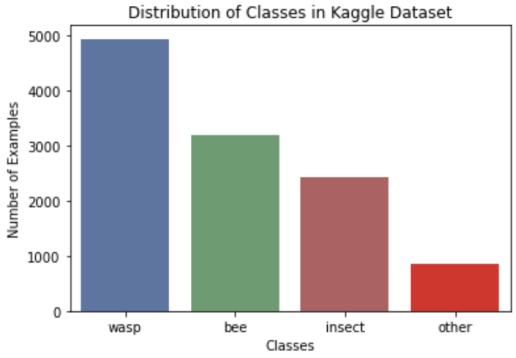


Figure 3. Class distribution of the Kaggle dataset

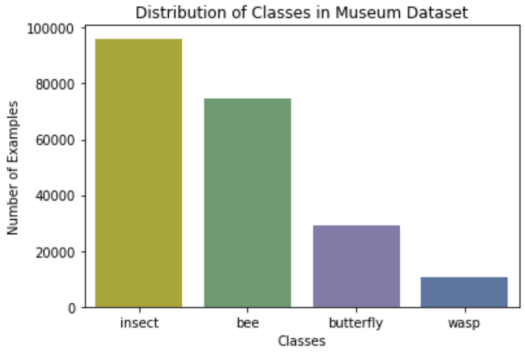


Figure 4. Class distribution of the museum dataset

Each dataset has a missing class, i.e. 'butterfly' in Kaggle dataset and 'other' in Museum dataset, the combination of the two datasets results in total number of 5 classes.

To ensure a robust generalization performance, instead of the conventional **Train/Test split**, the Kaggle dataset, with its different source is used as the test set and the Museum dataset for the training set. Using the method established in [5], the main idea being one needs to see enough errors (about 100 as a rule of thumb) to get a small error bar (not seeing any error would imply low precision), **N = 100/E**. Assuming an error-rate of about 10%, we thus have kept a test set of 12,500 examples, and a training set of 210,676 examples, and a separate validation set with 600 examples per class.

To make the competition fair, we have separeated the Museum Data from Kaggle i.e. in Train Dataset, there are no images from Kaggle dataset. On the other hand the Validation and Test datasets are made from Kaggle Dataset and some images are taken from Museum to complete the total number of images in these datasets. The other class in Train is taken from another source and some images from it are included in the Validation and Test set.

Table II shows the classes and number of images in Kaggle and Museum Dataset and Table III shows the number of images in each class in each dataset (train, validation and test).

Table II. Number of Images in each class in Kaggle and Museum

Dataset	Bee	Wasp	Butterfly	Insect	Other	Total
Kaggle	3182	4941	-	2439	853	11,415
Museum	74,413	10,796	29,361	96,106	-	210,676

Table III. Number of Images in each class in train,validation and test

Dataset	Bee	Wasp	Butterfly	Insect	Other	Total
Train	73,773	9,820	26,255	95,442	36,266	241,556
Validation	600	600	600	600	600	3000
Test	2500	2500	2500	2500	2500	12,500
Total	76,873	12,920	29,355	98,542	39,366	257,056

In the museum data we have some meta-data which is described in Table IV below.

Table IV. Meta-Data with Museum Data

URL	picture file URL
YMD	date in YYYY-MM-DD format
Lat	Latitude in decimal degrees
Long	Longitude in decimal degrees
Flower	flower taxa as from proposed Spipoll flower taxa list
collection	set of 520 mn pictures of arthropods on specific flower specie
user-id	User id
B&W-taxonomy	Taxonomy i.e. Bees, wasps, butterflies and Other insects
Nom-taxon	spipoll taxa name
ORDRE	hierarchical taxonomical level "order" (no ranking)
IFOR	hierarchical taxonomical level "infra-order"
SPFM	hierarchical taxonomical level "super-family"
FM	hierarchical taxonomical level "family"

Images in both datasets were not in uniform size and resolutions. Some images were captured in portrait mode and some in landscape. All images are processed as follows:

- Getting the center of the image and then cutting the top and bottom or left and right of the image depending on if the image is portrait or landspace. In this step the image is converted into a perfect square
- We have used opencv in python to resize the square image in a uniform resolution i.e 128x128

The museum dataset has images from different users and different camera devices so we have shown in Figure 5 the average sizes of images in each class. Figure 6 shows the storage size in GigaBytes the images in each class will take. Figure 7 shows the distribution of heights and widths of images in the museum dataset.

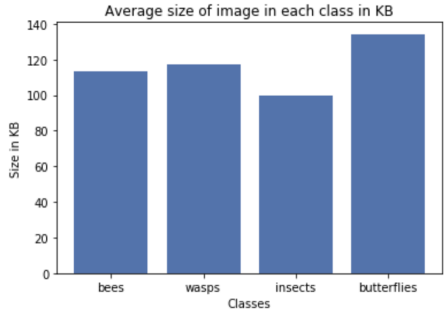


Figure 5. Average size per image per class in Museum dataset

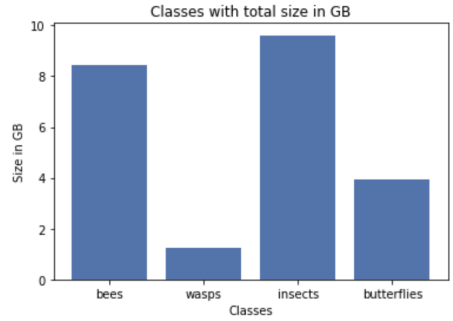


Figure 6. Storage size per class in Museum dataset

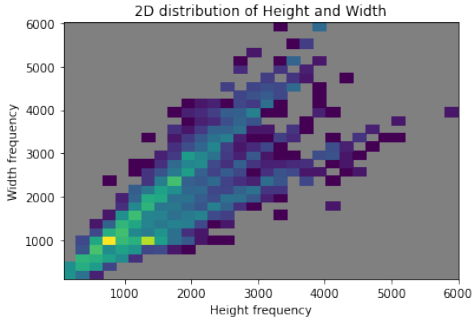


Figure 7. 2-D distribution of height and width of images in the dataset

The reason for choosing 128x128 for the initial phase is due to the large amount of data and therefore to make it manageable. A comparative analysis of resolutions is being done to quantify error rate with resolution changes might be made accordingly (see Figure 7 for a frequency distribution of image height and width).
See Figure 8 and 9 for unprocessed and processed images from museum data-set and figure 10 and 11 for Kaggle data-set respectively.

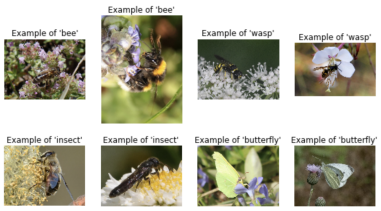


Figure 8. Raw samples from Museum dataset

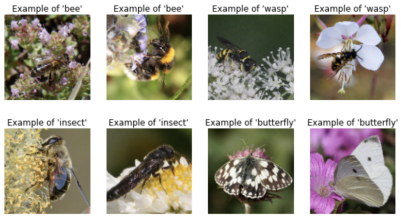


Figure 9. Pre-processed samples from Museum dataset

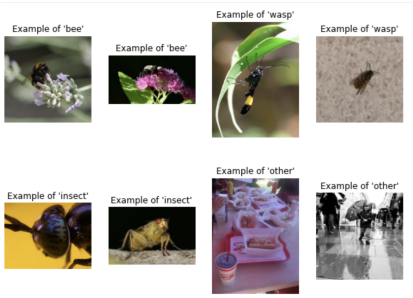


Figure 10. Raw samples from Kaggle dataset

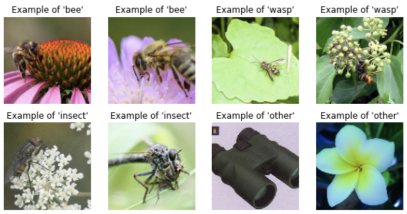


Figure 11. Pre-processed samples from Kaggle dataset

As the dataset is too big to be easily used, we have to reduce the resolution to reduce its total size. In order to do so, we are showing same examples from each class in different resolutions to see if we had any aliasing (See Figure 12-15).

Figure 12. Sample images with original resolution resized into square

Figure 13. Sample images with 256x256 resolution

Figure 14. Sample images with 128x128 resolution

Figure 15. Sample images with 64x64 resolution

For the 256x256 and the 128x128 we have no problems as the quality seems to be as good as the original resolution. However, when reducing the size to 64x64 we start to have some aliasing.
To corroborate our first impressions, we used the architecture "ResNet152V2" to get some results with the different resolutions but with the same number of examples for each (See Figure 16).

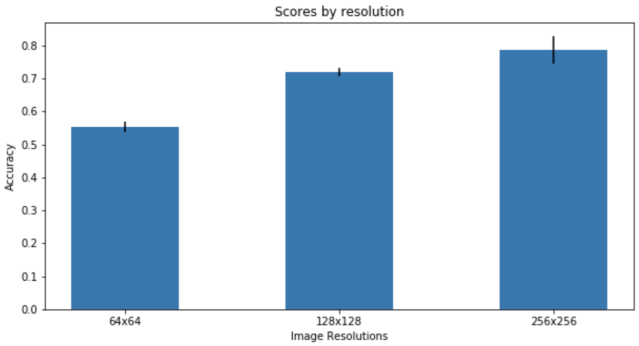


Figure 16. Score per resolution using ResNet152V2

For the 64x64 resolutions the aliasing seems to induce a big loss in the score compared to the rest of the resolutions so we discard this resolution for our dataset. On the other hand, between 128x128 and 256x256 we don't have a big difference in term of accuracy, especially when gain in accuracy is just 0.1 for a 16 times bigger dataset. Furthermore, we could add more examples to catch up this loss and still have a lighter dataset as compared to the dataset with 256x256 resolution images. Ultimately we keep the 128x128 resolutions as this choice allows us to add more examples in our dataset with minimal loss.

But this reduction also comes with a certain problem. When we applied the resolution reduction we didn't use any reduction filter, so we suspect our images contain aliasing. There are not many aliasing metrics so we used one of the only metrics that we could find, from a team from the Max Planck Institute of Berlin [15]. The paper makes use of a CNN to predict the areas of the image affected by aliasing. We need to give the set of images reduced with anti-aliasing filter (here we use OpenCV to make it) and the same images but without the anti-aliasing filter. Then the CNN will return the areas (a mask) where higher value means the aliasing is more likely present.

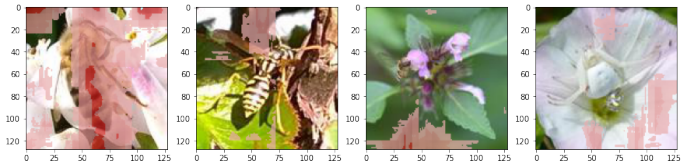


Figure 17. Aliasing detection in different images : the darker the red spots are, the higher the probability of aliasing in this area

This method of catching the aliasing is only visual. A lead for future work would be to compare the effectiveness of a model with two datasets : one with aliasing and the other without aliasing to observe its impact on prediction. In the case where the aliasing images gives a better score, we will use the OpenCV function 'resize' with an appropriate filter for the parameter 'interpolation'.

The distribution of the classes in our data is not uniform as shown in Figure 3, 4 (as is in real life problems) but our validation and test set has equal number of classes (see Table III). Therefore, to assess the performance, we use **Accuracy** metric to measure the performance of the model.

III. Preliminary Results

A. Baseline

We have preliminary results using a **Gaussian Naive Bayes** with OpenCV features extracted from images using KAZE descriptor [13] The model accuracy is not good and specially the validation and test accuracy is lower the train so it cannot generalize well. We observe that the epoch does not contribute in change in the score.

The KAZE descriptor is an algorithm in Open-CV to detect and describe multi-scale 2D features of images. These features, which can be specific points, edges, objects or shapes, can be used to identify specific elements or objects in an image. Consequently, the KEAZE descriptor can be used to extract key-points, feature matching between different images or object recognition. Thus can be used for image classification.

What makes this descriptor different from the others like SIFT, ORB, BRIEF etc is the scale space on which it is based. Most of the methods rely on Gaussian Scale Spaces, whose main drawback is the fact that they blur edges in images and smoothen noise and details at the same level, reducing localization accuracy and distinctiveness. In case of KAZE descriptor, the scale space is a nonlinear one obtained using diffusion filtering, a task with higher computing costs but leading to better results in terms of description and detection. In addition, we have evaluated some Scikit Learn models with the Train set, Validation set, Test set and a Cross Validation method. As you can see in Figure 18 the results are not very good for the test and validation score. The difference between the Train score and the Validation/Test score are huge. Especially, we can point the overfitting of RandomForestClassifier and a little bit of overfitting from MLPClassifier.

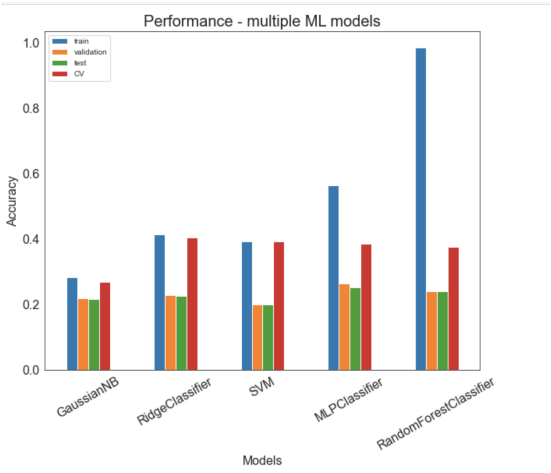


Figure 18. Comparison of different scikit-learn models

Furthermore, we have used a small subset of our training data and structured it according to the requirements [10] of AutoDL challenge [11] and submitted it to the AutoDL challenge [11] which is not like othe Codalab challenges which uses a specific dataset to evaluate the model of participants, but in AutoDL the winner AutoDL solution is applied on the data provided and it gives the opportunity to evaluate the model trained by DeepWisdom. We prepared two datasets: one with image pixels as features in tabular form and the second with images in its raw and pre-processed form with 128x128 resolution. The results obtained for tabular data can be seen in Figure 19 and for raw images in Figure 20. We can see a clear difference between the two and conclude that deep learning works better on raw images rather than tabular features.

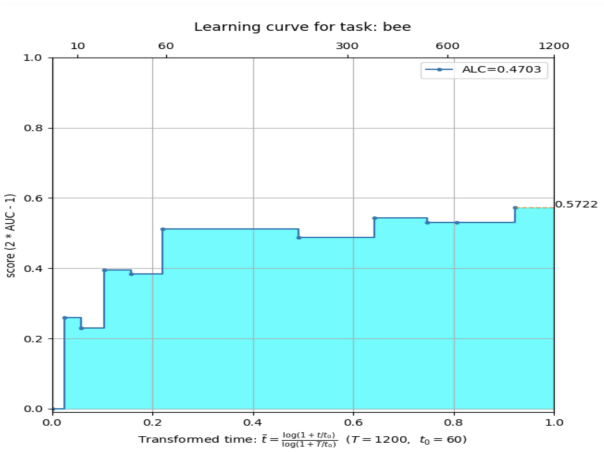


Figure 19. Baseline results from Auto-DL with tabular data with image pixels as features

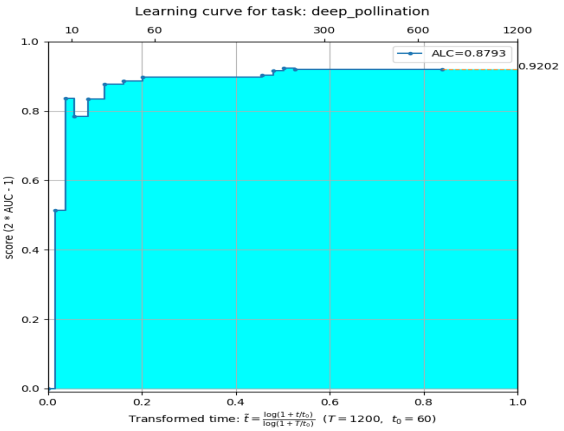


Figure 20. Baseline results from Auto-DL with raw images as data

B. High Performance

As discussed before, using deep pre-trained models for this task hasn't been done before [4]. This is a popular approach, known as transfer learning, where a model trained for one task is retrained for a second task and therefore exploiting the knowledge acquired during the training process on this first task. For this reason, and to demonstrate that better results that the ones achieved by the baseline models are feasible, a high performance model using transfer learning was developed.

As a preliminary exercise, we took a subset of the data consisting of 10,000 samples and trained using popular CNN architectures for image classification. Figures 21 and 22 below shows the results.

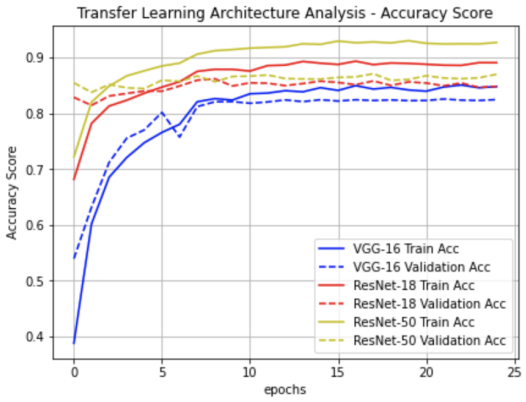


Figure 21. Training accuracy results for different architectures with transfer learning on a subset of the data

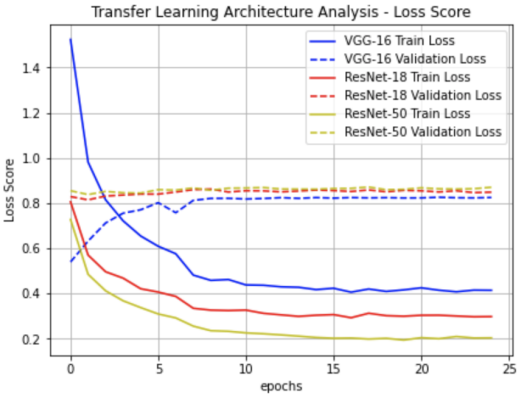


Figure 22. Training loss results for different architectures with transfer learning on a subset of the data

We can see that there is a stronger accuracy learning curve for VGG-16, while for ResNet-50 and ResNet-18 it is not as prominent. Although we can see some overfitting in the three architectures, it is fairly slight. However, when it comes to the loss, we can observe that the train loss decreases as expected over epochs but validation loss does not. For the two ResNet architectures it stays constant and, for VGG-16, it even increases for the first epochs before staying constant. This could be due to overfitting and the fact that, for Imagenet (what ResNet and VGG is trained on), there are no 'wasp' class and there is an 'insects' class, so this could lead to some loss increase.

The best results were achieved by ResNet-50 with a 87.13% validation accuracy score. It was trained on Cross Entropy Loss and Stochastic Gradient Descent over 25 epochs, with a batch size of 64 and a learning rate of 0.001. A learning rate decay by a factor of 0.1 every 7 epochs was used as well. This was developed using Pytorch and trained on Google Colab available (free) GPU. Overall, it took about 9 hours to train the model (45 minutes per epoch).

Hence, the following step was to use the ResNet-50 model to train using the entire dataset and evaluate its results. The results for this process can be seen in Figures 23 and 24 below.

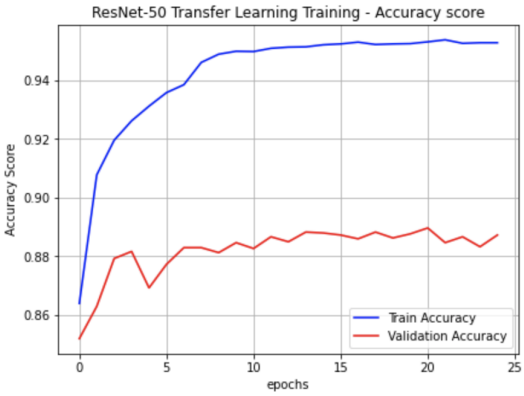


Figure 23. Training accuracy results for the ResNet-50 on the entire dataset

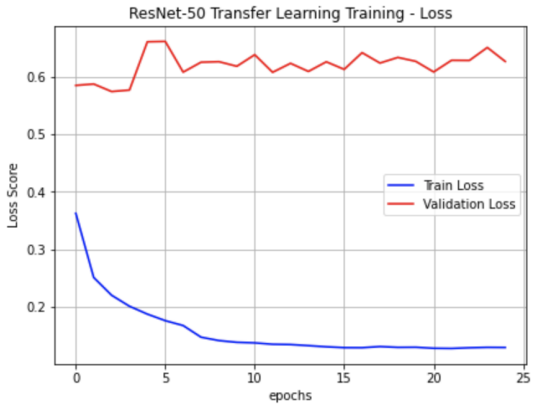


Figure 24. Training loss results for the ResNet-50 on the entire dataset

In this case, the best validation accuracy score was 88.96%. Although reasonable, we can see there is a slight overfitting. We then proceeded to evaluate the model with the test set, composed of 12500 samples (2500 samples per class). This yielded an accuracy of 87.82%, fairly close to the validation accuracy score. To evaluate how the model takes decisions, we computed a decision matrix for the model, which can be seen in 25.

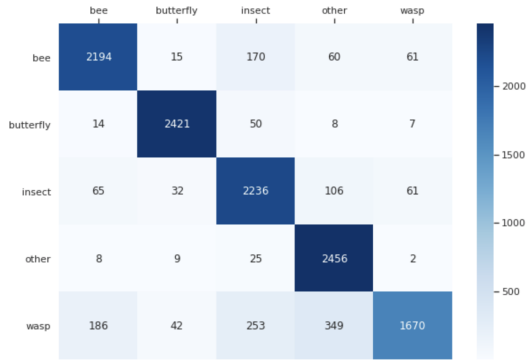


Figure 25. Confusion matrix for the ResNet-50 model on the test set

We can take some observations from this. The first thing that the matrix points out is that the wasp class is considerably misclassified with a higher degree than the rest, and more easily classified as 'others' or 'insect' than 'bee' or 'butterfly'. This makes sense since we have notoriously less 'wasp' samples (12,920) compared to the rest of classes, and we have almost ten times more samples of 'other's (98,542) than 'wasp's. Something interesting is that, although being the second shortest class in samples (29,355), the 'butterfly' class is the second best classified by our model. This can be due to the characteristic physical difference that butterflies present compared to bees, wasps and other pollinating insects. In the future, we could try to even out the number of 'wasp' samples to the rest of classes and observe if this misclassification decreases.

The difficulty level of the project can be observed not only from the preliminary results but also from another pertinent issue, Bias discussed in detail below.

C. Bias

To investigate bias in one of these datasets, we take same number of sample (randomly) from Museum and Kaggle data containing only **Bees** and **Wasps** to eliminate uncertainty because of the unbalanced number of images. In addition we split the experimentation for each class to locate the bias more efficiently. Our first approach is to train a strong Convolutional Neural Network to classify if an image is from Kaggle or Museum dataset. If we don't have any bias, we expect to have a balanced confusion matrix i.e approximately the same number for TP TN FP and FN.

We are using ResNet152V2 without any major pre-processing except which is required for the model, i.e. normalization of pixels. The accuracy obtained for wasps images is 0.91 and for bees 0.73. The confusion matrices are shown in Figure 25 and Figure 26.

		Actual class	
		Museum	**Kaggle**
Predicted class	**Museum**	**1593**	67
	Kaggle	214	**1388**

Figure 26. Confusion matrix with only Wasp

		Actual class	
		Museum	**Kaggle**
Predicted class	**Museum**	**843**	221
	Kaggle	346	**691**

Figure 27. Confusion matrix with only Bee

According to the accuracy score and the confusion matrix, the model can classify wasps from both datasets with little error but for bees the error is high as compared to the case of wasps. We observe that theere is a bias in at least one of the datasets.

To observe a fairly general difference between the two datasets, we compute the mean of each RGB component for each class in each dataset. The result obtained is shown in Figure 27 below.

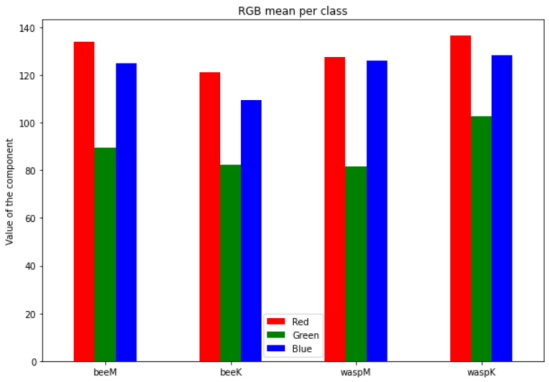


Figure 28. Mean of each RGB coponent for each class of each dataset. The letter K means Kaggle and M means Museum

As observed, bees images have the same average color but the brightness is not the same. Museum images are more luminous than the Kaggle images for the same color. The bias may be due to the camera used or the conditions in which the pictures are taken, which may differ. For wasps the average color is clearly not the same for the two datasets, Which could be a clue as to why the model finds them more easily.

We have made some important observation during the exploration of the Kaggle dataset and we have identified

some possible sources of bias:
Some samples from the Kaggle dataset were originally created with the intention to detect whether bees were carrying pollen or not. This makes that sets of samples very similar because of the similar blue backgrounds. Therefore, these sets of samples are not completely independent from each other as shown in Figure 29.

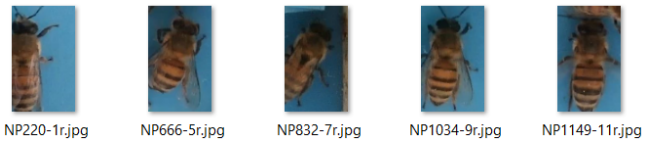


Figure 29. bee images from the folder /bee2 of the Kaggle dataset with the same background

The directory /bee2 of this dataset contains 714 images of bees with blue background. These kind of images are really problematic for the model because, the model can make correlations between the background and the label as in some famous cases made in [8].

On the other hand, some of the samples were created from videos shot at the entrance of a bee colony in June 2017 at the Bee facility of the Gurabo Agricultural Experimental Station of the University of Puerto Rico. The same is done for extracting some wasp images. This implies that the background is not always the same but context repetition can be dangerous for two main reasons. First, if we take random images for train and test set from this folder, the classifier could find the right label more easily due to the same background of some images. Secondly, the model can make a correlation between the background and label as well. Figure 30 shows samples extracted from the same video.

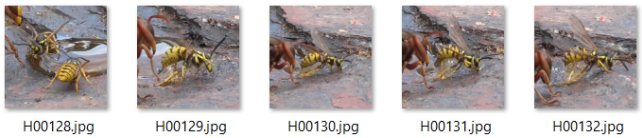


Figure 30. wasp images from the folder /wasp2 of the Kaggle dataset extracted from video

In order to address the issues highlighted in the experiments above, we have decided to remove half of the bees images and half of wasp images which contains either a similar blue background, in case of bee images, or a similar background because of the images extracted from a video. Although the number of images, after the removal of biased images, has decreases significantly but it will lead to a good and fair dataset provided to the participants of the competition.

IV. Explainability

Explainable AI (XAI) helps to understand the model's outputs for classification and regression tasks. To that extent, it can aid to verify that:

- the model is behaving as expected
- recognize bias in the model
- get ideas for ways to improve the model and also the training data

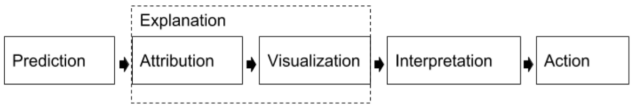


Figure 31. Chosen XAI workflow for the competition

For the purpose of a challenge creation, we particularly propose the use of XAI for the third point, i.e. to get a better glimpse into the data. Although there are multiple methods to attempt XAI, it depends on the type of the dataset and we focused on the post-hoc explanation method: feature attribution, which is best suited for image data. Feature attributions can indicate how much each feature in the model contributed to the predictions for each given example. Following are used commonly for feature attributions:

- Sampled Shapley values based methods (for tabular data)
- Gradient Based methods (mainly for image data, applicable only to differentiable models, such as neural networks, used especially for models with large feature spaces)

In essence, gradient-based methods for feature attribution propagate the prediction score, layer by layer of the deep network, from the output of the network back to its input, in the process assigning each pixel a score proportional to its importance. This gives us an opportunity to visualize the predicted class, along with an overlay of another image (a heatmap) showing which pixels in the image contributed most strongly to the resulting prediction.

To that extent, we implemented GRAD-CAM [12], a member of the family of gradient based methods known for its simplicity and effectiveness. Simple because the entire method can be summarized as global-average pooling of the gradients of the predicted class with respect to the inputs (or rather feature maps of a specific CNN layer). Effective because unlike other techniques (for example, CAM), we don't need to re-train the model for just explainability and we can use the existing model as it is. We used VGG-16, pre-trained on ImageNet, and trained its last convolution block (block5_conv) along with the final layers on our complete dataset of 241,556 images, freezing rest of the layers (the training time was 6 hrs on average for 10 epochs). The choice lies in the fact that VGG-16 is amongst the simplest in architecture which still is known to deliver good performance, thus easier to fine-tune.

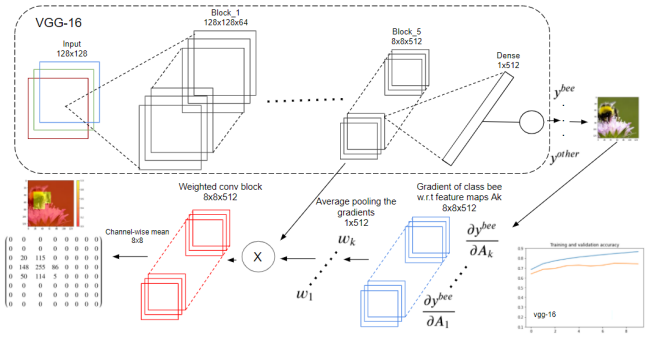


Figure 32. Working of Grad-CAM

The gradients of the predicted class score c are calculated with respect to the input. In our case, it is with respect to feature maps A_k of the last convolution block 'block5_conv' of the VGG16 used,

$$\frac{dy^c}{dA^k_{i,j}}$$

measures linear effect of $(i,j)^{th}$ pixel point in the k^{th} feature map on the c^{th} class score. So averaging pooling give us weights w^c_k of this gradient across i and j captures the importance of feature map k on the c^{th} class score.

$$w^c_k = \sum_i \sum_j \frac{dy^c}{dA^k_{i,j}}$$

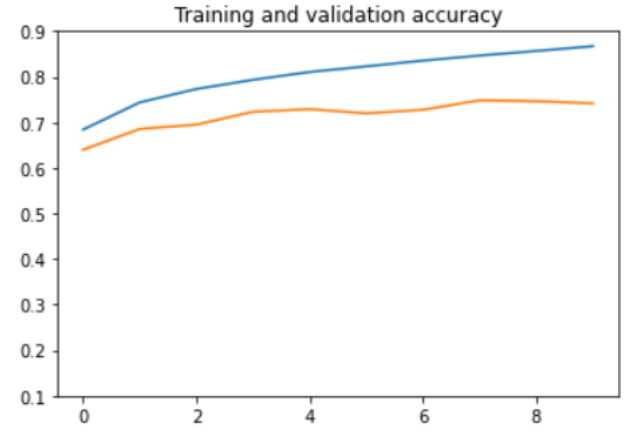


Figure 33. Performance of VGG-16 based on transfer learning used for Explainability

Quantifying explainability for the competition

As organizers of this competition, where we've proposed $Explanability$ as one of the tasks, a non-trivial issue is then to quantify how well the challengers perform in the this task. The existing methods in the literature for image data XAI are instance based and qualitative in nature.

Inspecting the feature attributions for an individual prediction may provide good insight, but the insight may not be generalizable to the entire class for that individual instance, or the entire model. To get more generalizable insight, there then seems to be a need to aggregate attributions over subsets over the entire dataset. But this is in effect contrasting to the nature of the method deployed, which offers instance level feature attribution. Therefore, we encourage the participants to improve their models via getting a score for correctly annotating objects in the image using Grad-CAM or other feature attribution methods for images.

Ideally, a model should give importance to features where the class is present and not the objects around it, else its predictions are not trustworthy. Many authors have previously used humans to assess the trust worthiness. Given the nature of the challenge, it is not feasible to have human evaluators in the loop. Yet the trustworthiness of the model can be assessed if the model makes its decisions based on features relevant to the object, and not other factors in the background.

Therefore, we challenge the participants to find the center of their feature attributions, which should be close to the center of the hand-annotated bounding-boxes, quantifying good feature attribution. We hand-annotated a small subset of the train dataset (10 images per class) to form bounding boxes around the class instance for all the subsets of the dataset. We carefully picked instances where the object is not in the center but rather around the edges of the image, to avoid a heuristic prediction of the center.

In figures 34-59, we visualize the overlayed images of the feature attributions and the original image, which gives us much insight into the model and the data, and we see some examples where the class attribution is incorrect and the model detects counterfactual objects in the background.

We use a soft bounding-box metric between the ground truth and the predicted center to assess the quality towards explainability. The participants should be able to achieve this, for example in the case of Grad-CAM via extracting the highest pixel corresponding to the highest intensity of the gradient of the predicted class with respect to the output of the last convolution block.

$$\exp\left(-0.5772 * \left[\frac{(x-x0)^2}{a^2} + \frac{(y-y0)^2}{b^2}\right]\right)$$

where (x_0, y_0) are the coordinates of the center of your bounding box, (x, y) the coordinates predicted by the participant, and a and b are the length and height of the bounding-box. This represents a score whose sensitivity or gradient is a factor of the size of the object in the image, lower for larger images, thus the idea of a soft bounding-box.

The model we used for the purpose of XAI is VGG-16 trained on the entire dataset with a test accuracy of 78.3%. We provide it as a a baseline model alongwith relevant code of Grad-CAM for the Explainability task. The participants would be expected to create a csv in a particular format with predicted center of objects which would be scored using the methodology thus defined.

Finally, as can be seen from the figures, visualizing feature attributes brings effectiveness in understanding the data and model and we propose to investigate further as a future scope of our work.

A. *Wrong Predictions*

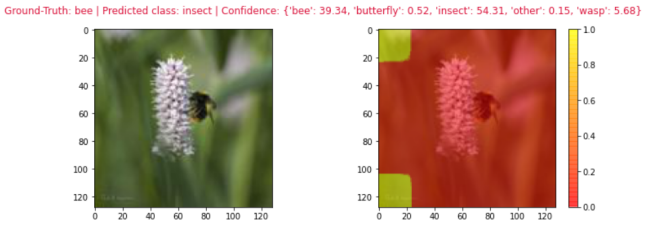


Figure 34. Example of a counterfactual in the image. The network detects a signature on the bottom-left

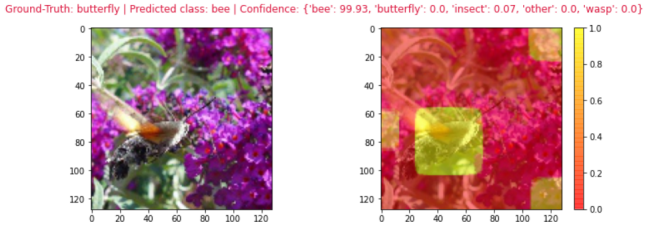


Figure 35. Correct class attribution, but butterfly is incorrectly classified as a bee

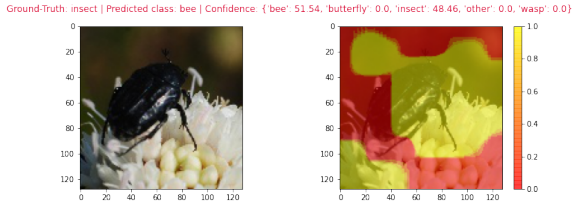


Figure 36. Correct class attribution. Variability of the insect class might be a problem

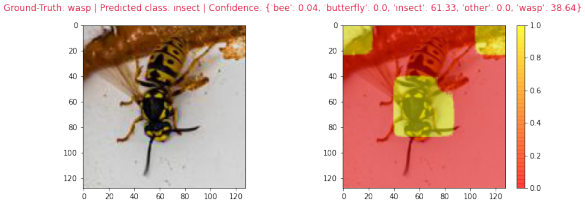


Figure 37. Correct class attribution, but wasp is incorrectly classified as an insect

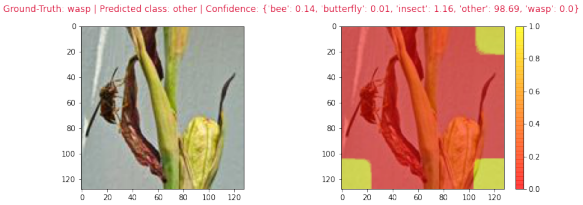


Figure 38. Wrong class attribution. The background has higher feature attribution in this example

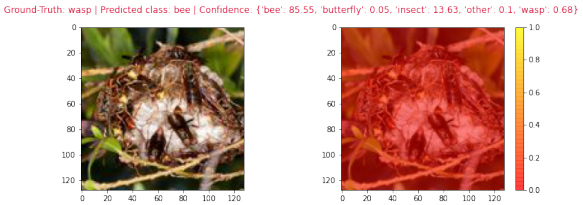


Figure 39. No class attribution. Multiple wasp objects in the image might be a reason

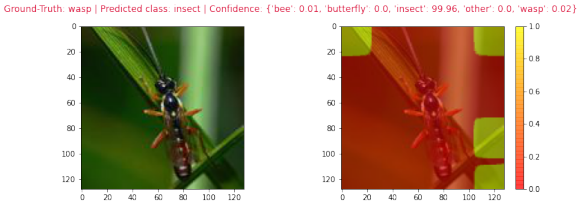


Figure 40. Incorrect localization of the class

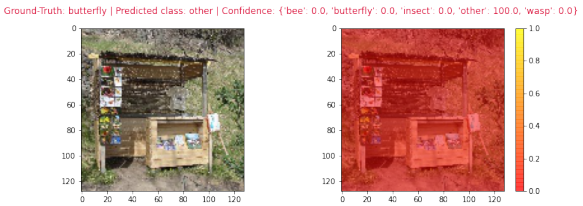


Figure 41. Possibly a mistake in the labeling since visual inspection deems support to the 'other' class

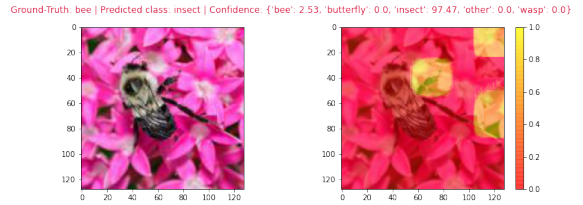


Figure 42. Correct class localization but variability in the 'bee' object might be an issue

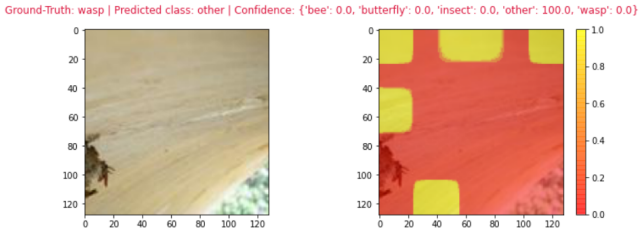


Figure 43. Class not localized although it is barely present

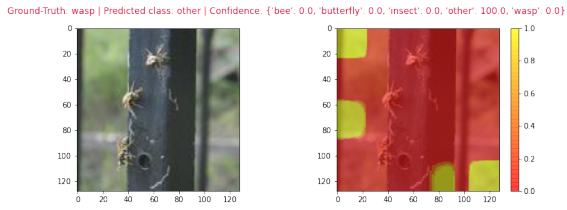


Figure 44. Incorrect class localization. Background has higher feature attribution

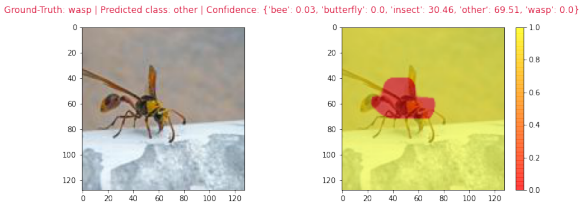


Figure 45. Surprising inverted class attribution

B. *Good predictions*

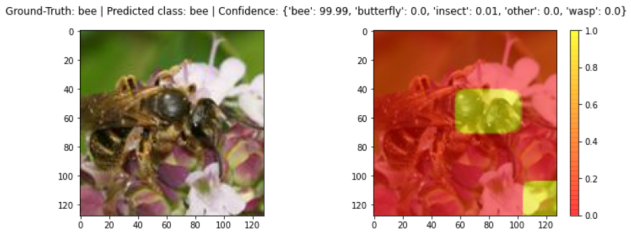


Figure 46. Correct class attribution as a bee

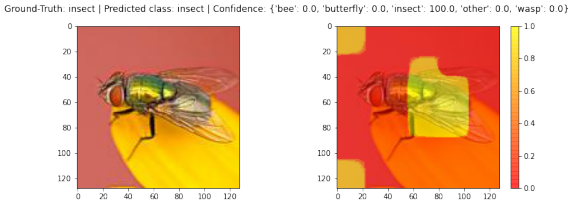


Figure 47. Correct class attribution as an insect

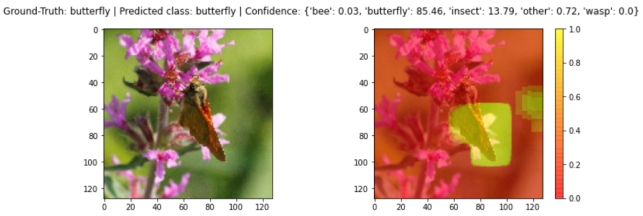


Figure 48. Correct class attribution as a butterfly

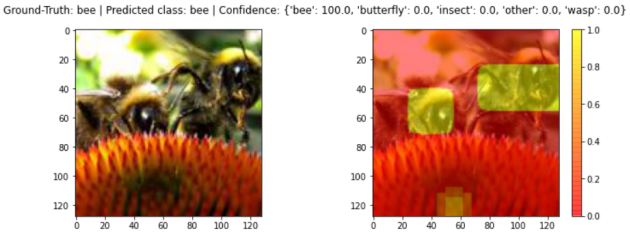


Figure 49. Correct class attribution of multiple bees in the image

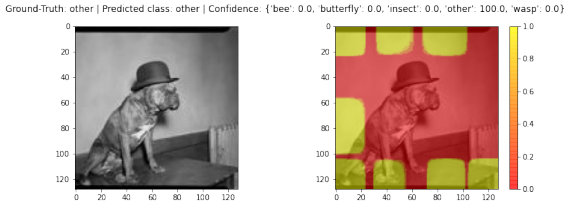


Figure 50. Correct prediction. To be noted that the model is based on Transfer Learning with ImageNet weights for the base layers

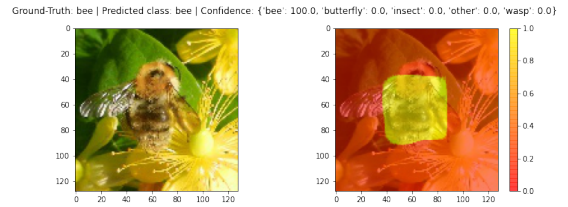


Figure 51. Bee class predicted with high confidence. Correct class attribution

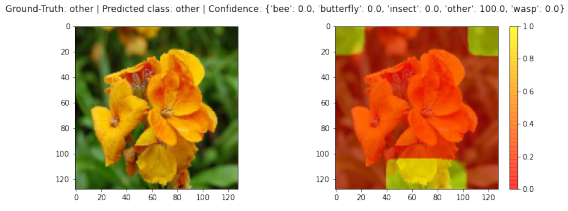


Figure 52. With flowers in the image, 'other' class predicted with high confidence

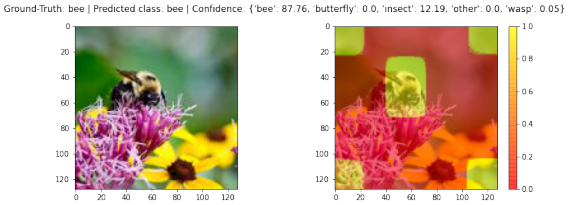


Figure 53. Correct class attribution for bee. Background has some feature attribution too as can be seen from the confidence level

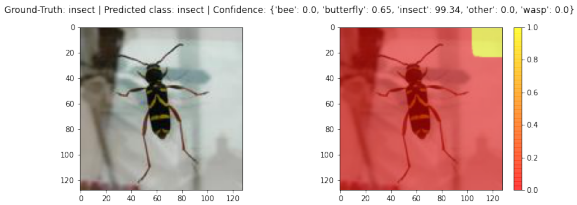


Figure 54. Incorrect class attribution, but correct prediction

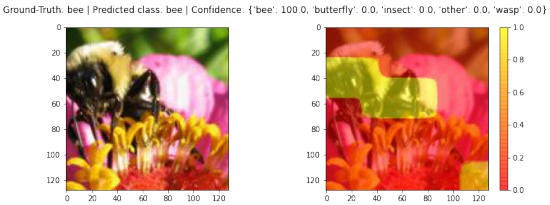


Figure 55. Correct class attribution for multiple bee class objects

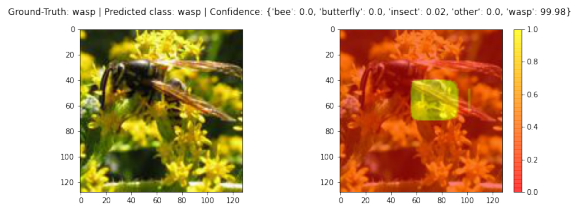


Figure 56. Correct class attribution as wasp

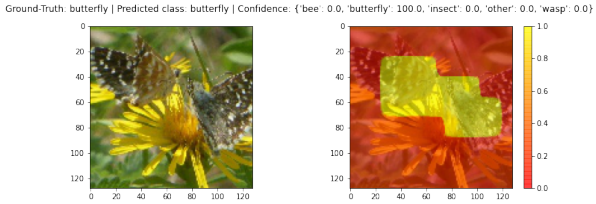


Figure 57. Correct class attribution for multiple butterfly class objects

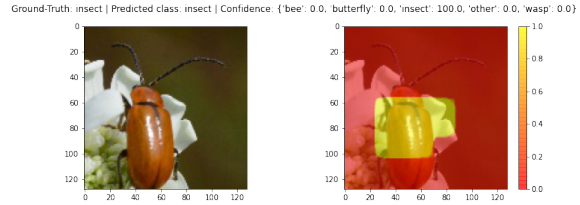


Figure 58. Correct class attribution for insect class objects

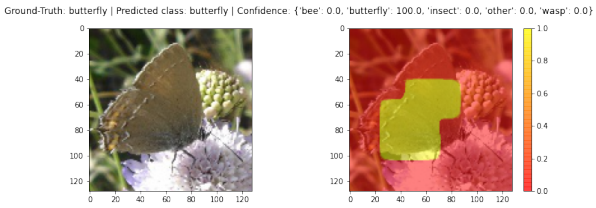


Figure 59. Correct class attribution for a butterfly in the image

Corresponding SIFT keypoints: Another method we employed was to extract SIFT keypoints and find correspondence between different images of the same class to understand if there are certain representative features for a particular class. What we found was that using SIFT features was not a viable option, as can be seen from Figure 60, 61, for same images from different angles, not many keypoints were found. This should be expected since the the images are gathered in general settings and thus vary largely in background. Therefore the descriptors for each keypoint would not be able to capture generic features that would hold across different instances of the same class. It would be much better suited for images gathered from lab-based settings (same background), which is not the case with our dataset.

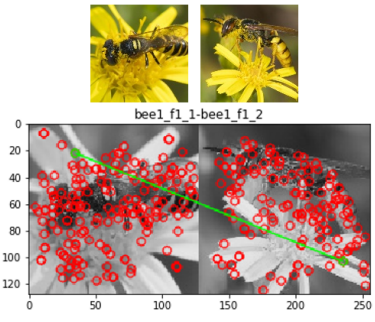
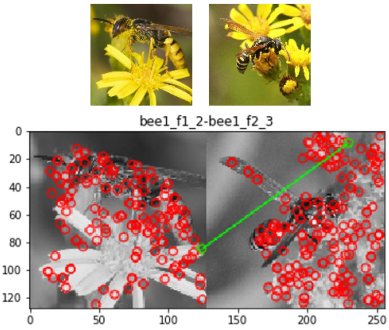


Figure 60. Corresponding SIFT keypoints of similar examples



[H] Corresponding SIFT keypoints of similar examples

Layer feature visualization: One final suggestion we offer to the challenger is to visualize the output of intermediate layers to become comfortable with a CNN model rather than a blackbox. In Figure 63 and 65 using images shown in Figure 62 and 64 respectively, each row is the output of a layer, and each image in the row is the output of a specific filter in that output feature map. We see that we go from raw pixels of the images to increasingly abstract representations. Note that some images are blank due to dropout being employed and set to 0.5. The representations downstream highlight what the network pays attention to and they show fewer and fewer features being "activated", most are set to zero. These representations carry increasingly less information about the original pixels of the image, but are supposed to represent increasingly refined information about the class of the image.

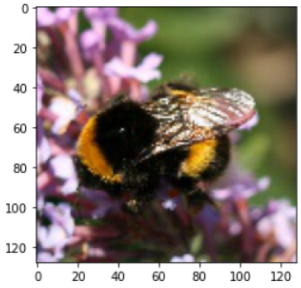


Figure 61. A random example

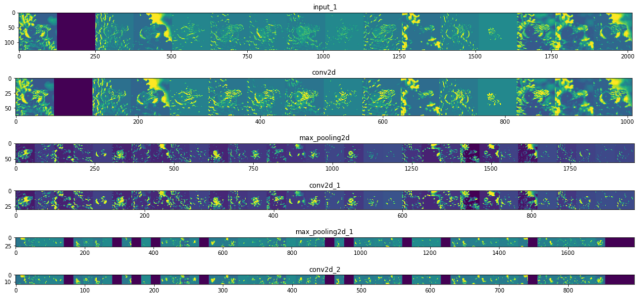


Figure 62. Features extracted by each convolutional layer - Explainability

Figure 63. A random example

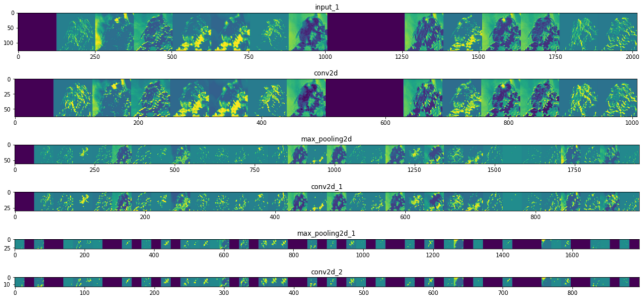


Figure 64. Features extracted by each convolutional layer - Explainability

V. Conclusion

In conclusion, we gathered the data from Kaggle and the Museum, then we worked on finding out a bias between the two datasets and we were able to detect one with the luminosity and the colors. We later found another one in the Kaggle dataset with some bees being only shown on specific backgrounds and images created from videos shot. Working with the model VGG-16 and the GRAD-CAM we got a first insight on how our model worked to classify the pictures. Thanks to this primarily work, we were able to create a challenge with two versions: One with data consisting of openCV features extracted from images and Second with raw images as the input data. The whole process of creating and organizing challenge was very interesting and there were a lot of things to learn including Data collection, Data processing, Feature Extraction, Bias and Defects in Data and the most important part to present the challenge to participants in a very interesting way using Codalab. Another interesting part of the project was that, unlike normal Machine Learning problems, we gained insight into the predictions of the machine learning model via Explainability. We introduced an ad-hoc metric for Explainability to quantify the discrepency between annotated object's center in the image and predicted center via feature visualization technique (Grad-CAM), suitable for the purpose of competition. It was interesting to not only provide participants with a baseline model to help them get started but also solving our own challenge and achieving the best

possible performance in order to determine the limits of our dataset and to set a high leaderboard score. We hope that this competition will be one of the interesting competitions which our participants have participated in.

VI. Acknowledgements

We thank the team at the Museum National d'histoire Naturelle for giving us access to their huge dataset of insects, which made our inquiry possible. We are also indebted to our mentor Professor Isabelle Guyon for her careful supervision, kind support during the project and giving us access to the CodaLab platform. We are also thankful to Zhengying Liu, Michael Vaccaro and Adrien Pavao for helping us with our inquiries along the way, and finally to Google for giving us access to the resources of the Google Cloud Platform.

VII. References

[1] Weisser, W.W., Siemann, E. (2008), The Various Effects of Insects on Ecosystem Functioning. In: Weisser W.W., Siemann E. (eds) Insects and Ecosystem Function. Ecological Studies (Analysis and Synthesis), vol 173. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74004-9_1

[2] Rodriguez, I. F., Megret, R., Acuna, E., Agosto-Rivera, J. L., Giray, T., (2018), Recognition of Pollen-Bearing Bees from Video Using Convolutional Neural Network, IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, pp. 314-322, https://ieeexplore.ieee.org/document/8354145

[3] Russakovsky, O., Deng, J., Su, H., Krause, J, Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Fei-Fei, L., (2015), ImageNet Large Scale Visual Recognition Challenge, https://arxiv.org/abs/1409.0575

[4] Martineau, M., Conte, D., Raveaux, R., Arnault, I., Munier, D., Venturini, G., (2017), A survey on image-based insect classification, https://hal.archives-ouvertes.fr/hal-01441203/document

[5] Guyon, I., Makhoul, J., (1998), What Size Test Set Gives Good Error Rate Estimates?, IEEE Transactions on Pattern Analysis and Machine Intelligence, https://ieeexplore.ieee.org/document/655649

[6] Goksel, K., Carmichael, E., Ramaswami, A., (2017), AutoML Data Format, Github, https://github.com/codalab/chalab/wiki/Help:-Wizard-%E2%80%90-Challenge-%E2%80%90-Data/

[7] Ullah, I., Lansari, M., Bou, X., Soulard, T., Arora, V., Santiaga Garcia, E., (2021), Project Deep Pollination Starting Kit, Github, https://github.com/Ecologists/deep_pollination/

[8] Condliffe, J., (2020), AI Learns Sexism Just by Studying Photographs, MIT Technology Review, https://www.technologyreview.com/2017/08/21/149585/ai-learns-sexism-just-by-studying-photographs/

[9] Amit, Y., (2002), 2D Object Detection and Recognition. Models, Algorithms and Networks, The MIT Press, https://doc.lagout.org/science/0_Computer%20Science/2D%20Object%20Detection%20and%20Recognition.pdf

[10] Liu, Z., (2020), Instructinos to format data for AutoDL, Codalab, https://github.com/zhengying-liu/autodl-contrib

[11] Vaccaro, M., Pavao, A., Guyon, I., (2020), AutoDL Challenge, Codalab, https://competitions.codalab.org/competitions/27082

[12] Ramprasaath, R., Cogswell, M., (2019), Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, https://arxiv.org/pdf/1610.02391.pdf

[13] Fernandez Alcantarilla, P., Bartoli, A., Davison, A., (2012), KAZE Feature extraction, https://www.doc.ic.ac.uk/~ajd/Publications/alcantarilla_etal_eccv2012.pdf

[14] Kégl, B., Boucaud, A., (2018), The RAMP framework: from reproducibility to transparency in the design and optimization of scientific workflows, Boucaud Gramfort et al, https://openreview.net/pdf?id=Syg4NHz4eQ

[15] Wolski, K., Giunchi, D., (2017), Dataset and metrics for predicting local visible differences, https://visibility-metrics.mpi-inf.mpg.de/files/wolski2018local_metric.pdf